Music Lesson Service Application

Noah Abrams and Malik Saafir

Table of Contents

Analysis

Sources Used

Our sources were email interviews with Joseph Reed and Akirah Renee, two music instructors. In the interview the two instructors discussed features such virtual lessons (due to the current pandemic), social media connections for instructors, email lists, payment handling and reminders for students to keep up with payments, and flexible pricing.

Features List:

Instructor search: Users can search for instructors based on information such as instrument(s) of expertise, availability during a specific date range, and pricing rates. Instructors in the search results will show up with more information like their bios, instrument(s) of expertise, social media links, etc.

Reviews: users can leave reviews on instructors after receiving lessons. This can also factor into the search feature (sort results by rating).

Lesson booking: users can book lessons with instructors (only allowing virtual lessons due to the current pandemic). Users can get reminders to their email they used to sign up so they keep track of lessons coming up.

Payment: The site lists the rates instructors charge and reminds users to pay when it is due.

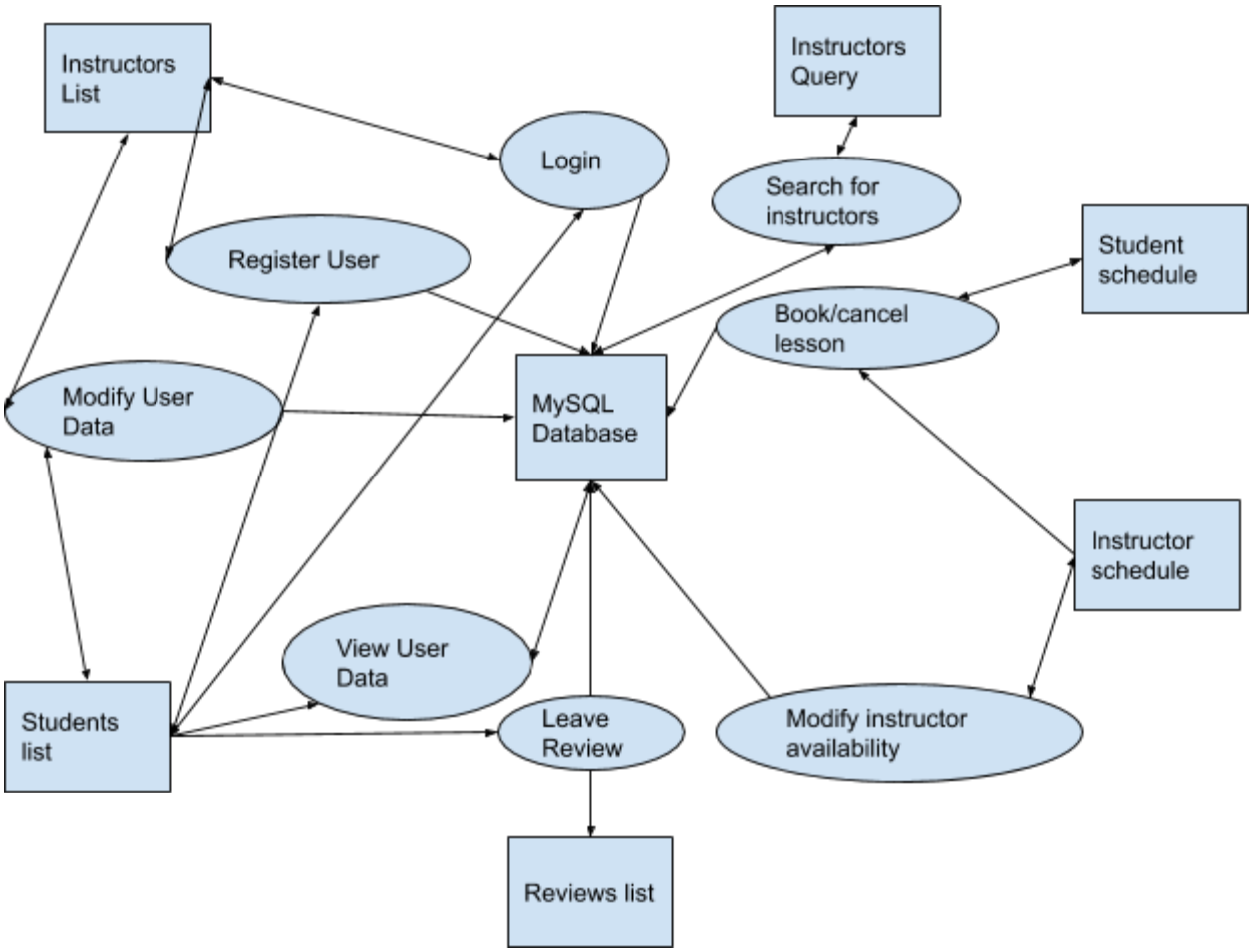
Registration / login for students and instructors: when an instructor logs in they can see and edit their information and availability schedule and accept lessons. They can also add links to external instruction materials of their choice, such as Youtube channels or personal websites. When a student logs in they can see upcoming lessons, register for lessons, search for

instructors, see upcoming payments due, and see and modify their own information.

Prioritized feature list:

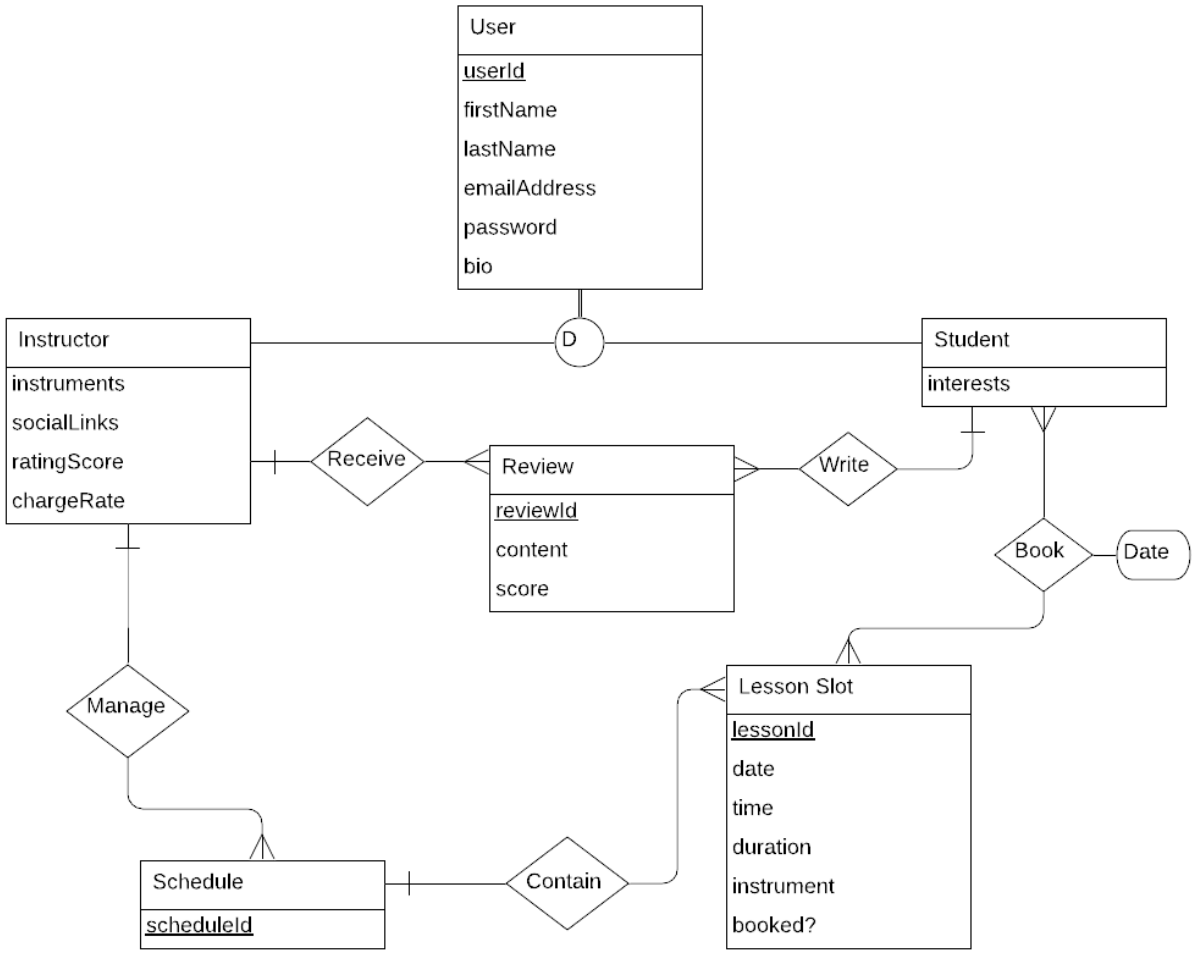
1. Registration / login (without this nothing else will function)
2. Instructor search
3. Lesson booking
4. Reviews
5. Payment

Information Flow Diagram



Specification

EER Diagram:



Task Forms

Task Name: Enter Student

Task Number: T01

Description: takes string input of student's name, phone number, email, and their instrument(s)

Enabling Condition: All fields entered and validated

Frequency: 30+/day

Input: ED: Students

Ent.: Students, User Info

Rel.: Registers

Output: Student is registered for web service

Operations:

```
DO
     Print("Enter your information: ")
     Read(FirstName, LastName, EmailAddress, Password,
     VerifyPassword, PhoneNumber, Instruments)
WHILE (Password != VerifyPassword)
SaveToDatabaseStudentsList(FirstName, LastName, EmailAddress,
Password, PhoneNumber, Instruments)
Print("Welcome, ", FirstName, LastName, "!!!")
```

Subtasks: Show user initial compatible instructors

Task Name: Register Instructor

Task Number: T02

Description: takes string input of instructor’s name, phone number, email, address, instrumental knowledge, and course offerings and prices.

Enabling Cond.: All fields entered and validated

Frequency: 10+/day

Input: ED: Instructors

E: Instructors, User Info

R: Registers

Output: Instructor is registered for web service

Operations:

```
DO
     Print(“Enter your information: “)
     Read(FirstName, LastName, EmailAddress, Password,
     VerifyPassword, PhoneNumber, InstrumentalExperience,
     ChargeRate)
WHILE (Password != VerifyPassword)
SaveToDatabaseIntsructorsList(FirstName, LastName, EmailAddress,
Password, PhoneNumber, InstrumentalExperience, ChargeRate)
```

Subtasks: Enter User Info

Task Name: Instructor Modify Lesson Schedule

Task Number: T03

Description: Takes in string input of the name, time, date, duration ,instrument played during, and difficulty of a new lesson session being posted; takes in int input to confirm cancellation of posted lesson

Frequency: 7+/day

Input: ED: Instructors

E: Instructors, Schedule

R: Manage

Output: “Instructor your [session name] session for [session time] has been posted”; “Instructor your [session name] session for [session time] has been deleted”

Operations:

```
Print(“Would you like to add or remove a lesson slot?”)
Read(Choice)
If Choice == “add” then
     Read(SessionName, SessionTime, SessionDate,
     SessionDuration, SessionInstrument, SessionDifficulty)
     If DoesNotConflictWithSchedule(SessionTime, SessionDate,
SessionDuration) then
          SaveToDatabaseSchedule(SessionName, SessionTime,
          SessionDate, SessionDuration, SessionInstrument,
          SessionDifficulty)
     Else
          Print(“Conflicts with your current schedule”)
Else if Choice == “remove” then
     Read(SessionName, SessionTime, SessionDate)
```

```
    RemoveFromDatabaseSchedule(SessionName, SessionTime,
SessionDate)
Print(“Schedule has been updated”)
```

Subtasks: n/a

Task Name: Query Instructors List

Task Number: T04

Description: Takes in string as input session difficulty, availability, instrument, and preferred instructor review rating

Frequency: 100+/day

Input: ED: Instructors

E: Instructors Query, Instructor Schedule

R: Query availability

Output: “Here is a list of instructors matching your criteria” [list of instructors]

Operations:

Print(“Enter your search parameters: ”)

Read(SessionDifficulty, AvailabilityDateRange, Instruments, ReviewRatingRange, SortingCategory)

Results ← QueryInstructorList(SessionDifficulty, AvailabilityDateRange, Instruments ReviewRatingRange, SortingCategory)

DisplayResults(Results)

Subtasks: n/a

Task Name: Enter User Info

Task Number: T05

Description: Takes in string as student's name, phone number, email, password, username, and their instrument(s)

Frequency: 100+/day

Input: ED: Students

E: Students, User Info

R: Register Student, Login

Output: "Here is a list of instructors matching your criteria" [list of instructors]

Operations:

Instruments, SkillLevel ← LoadStudentInfo(StudentId)

Results ← QueryInstructorsList(Instruments, SkillLevel)

Print("Thank you for joining! These instructors match your skills and interests:", Results)

Subtasks: n/a

Task Name: Student Book/Cancel a Lesson

Task Number: T06

Description: Student books an available lesson with an instructor or cancels one they have signed up for

Frequency: 7+/day

Input: ED: Instructors

E: Student, Lesson Slot

R: Book/Cancel

Output: “Your lesson has been updated”

Operations:

```
Print(“Do you want to book or cancel a lesson?”);
Read(Choice);
If Choice == “book” then
     Print(“Which lesson do you want to book?”);
     Read(LessonChoice);
     If LessonAvailable(LessonChoice) then
          BookLesson(LessonChoice);
Else
     Print(“Which lesson do you want to cancel?”);
     Read(LessonChoice);
     CancelLessonBooking(LessonChoice);
```

Subtasks: updates availability of instructor

Task Name: Leave Review

Task Number: T06

Description: Takes in string as student’s name, instructor’s name, and comments, and an integer input for a rating score

Frequency: 8+/day

Input: ED: Reviews List

E: Students, Review

R: Query Reviews, Leave Review

Output: “Your review for [instructor’s name] has been submitted!”

Operations:

Read(StudentName, InstructorName, Comments, RatingScore)

SaveToDatabase(StudentName, InstructorName, Comments, RatingScore)

Print(“Your review for “, InstructorName, “ has been submitted!”)

Subtasks: n/a

Design

Relational Schema

Legend: **entity**, primary_key, *foreign_key*

User(user_id, fname, lname, email_address, password, is_teacher, is_student, charge_rate)

Instrument(instrument_id, name)

TeachesR(*teacher_id*, *instrument_id*)

Lesson(lesson_id, date, start_time, end_time, *teacher_id*, *student_id*, *instrument_id*)

Abstract Code for Tasks

1 Register New User:

/* read($fname, $lname, $email, $pword, $pword_verification, $teacher_or_student, $instrument_list, $charge_rate) */

/* check that $pword == $pword_verification */

/* if $teacher_or_student == teacher, then set $is_teacher = 1 and $is_student = 0, else set $is_teacher = 0 and $is_student = 1 */

/* 1.1 check that there is no user record with $email already */

SQL: select user_id from users where email_address = $email;

/* 1.2 if $email is not taken, insert data */

SQL / PHP: insert into users (fname, lname, email_address, password, is_teacher, is_student, charge_rate) VALUES ('$fname', '$lname', '$email', '$pword', $is_teacher, $is_student, $charge_rate);

/* insert teacher-instrument relations if $is_teacher == 1 */

/* 1.3 for each instrument in $instrument_list (counter variable $i): */

SQL / PHP: insert into teaches_r (teacher_id, instrument_id) values ({$user_id}, {$instruments[$i]}); // $user_id is the id of the inserted record

2 User Login:

/* read($email, $pword) */

SQL / PHP: select user_id from users where email_address = '$email' and password = '$pword';

/* if query returns 0 records email or password was incorrect, have user try again */

3 View Instructor List:

/* 3.1 print results as rows */

SQL: select concat(fname, ' ', lname) as teacher_name, email_address, user_id as teacher_id, charge_rate from users where is_teacher = 1 order by lname asc;

/* 3.2 for each row get the instruments the instructor teaches by teacher's user_id*/

SQL / PHP: select instruments.name as instrument_name from instruments natural join teaches_r inner join users on user_id = teacher_id where teacher_id = {$row['teacher_id']} order by instrument_name asc;

4 View Instructor's Lesson Availabilities:

/* for a chosen instructor view the lessons that are either open or already booked by the current user, displaying stored information as well as calculated duration and total lesson cost */

SQL / PHP: select lesson_id, instruments.name as instrument_name, date, start_time, end_time, charge_rate, round(timestampdiff(MINUTE, start_time, end_time) / 60, 2) as duration, round(charge_rate * timestampdiff(MINUTE, start_time, end_time) / 60, 2) as total_cost, lessons.student_id as student_id from users inner join lessons on users.user_id = lessons.teacher_id inner join instruments on lessons.instrument_id = instruments.instrument_id where

lessons.teacher_id = {$teacher_id} and (lessons.student_id is NULL or lessons.student_id = {$login_session['user_id']}) order by date asc, start_time asc;

5 Student Books Lesson:

/* 5.1 user picks a lesson to book, we check that the lesson doesn't overlap with any of the student's current lessons */

SQL / PHP: select lesson_id from lessons where student_id = {$login_session['user_id']} and date = '$lesson_date' and ((start_time between '$lesson_start' and '$lesson_end') or (end_time between '$lesson_start' and '$lesson_end'));

select date, start_time, end_time from lessons where lesson_id = {$_POST['lesson_id']};

/* 5.2 if there are no overlaps we book the lesson */

SQL / PHP: update lessons set student_id = {$login_session['user_id']} where lesson_id = {$_POST['lesson_id']};

6 Student Cancels Lesson:

/* student picks lesson they have booked to cancel, we update table */

SQL / PHP: update lessons set student_id = NULL where lesson_id = {$_POST['lesson_id']};

7 Instructor Adds Lesson Slot:

/* 7.1 present instructor with choice of instruments that they teach */

SQL / PHP: select instrument_id, instruments.name as instrument_name from instruments natural join teaches_r where teaches_r.teacher_id = {$login_session['user_id']};

/* read($lesson_instrument, $lesson_date, $lesson_start, $lesson_end) */

/* check whether start_time is before end_time, if not try again */

/* 7.2 check for overlaps in the instructor's current schedule */

SQL / PHP: select lesson_id from lessons where teacher_id = {$login_session['user_id']} and date = '$lesson_date' and ((start_time between '$lesson_start' and '$lesson_end') or (end_time between '$lesson_start' and '$lesson_end'));

/* 7.3 if no overlaps, add lesson availability */

SQL / PHP: insert into lessons (date, start_time, end_time, teacher_id, student_id, instrument_id) values ('$lesson_date', '$lesson_start', '$lesson_end', {$login_session['user_id']}, NULL, $lesson_instrument);

8 Instructor Removes Lesson Slot:

/* instructor picks one of their own lesson slots to delete */

SQL / PHP: delete from lessons where lesson_id = {$_POST['lesson_id']} and teacher_id = {$login_session['user_id']};

Implementation

See html, css, php, and mysql files:

- Run music_lesson_service.sql to create the database and insert starting data

Testing Instructions

- Open website in browser (it will automatically direct you to the login page if you have not logged in yet)
- Register a new account
- If you are a student account:
    - Click the link to view the instructors list
        - Press the register button to sign up for an instructor's class(es)
        - Book and/or cancel a class
        - Repeat as many times as you would like
    - View your own schedule
        - Cancel classes if you would like
- If you are a teacher account:
    - Go to your lesson schedule page
    - Add lesson slots with the instruments you have chosen
    - Delete lesson slots if you wish
- Log into a student account again and attempt to book any new slots you have created
- Test the login feature by logging out of your account and logging back into it
- Try to register a new account with the same email (it should not let you)